Lab 16 - Classes

**Questions:**

**1.** Try to answer this question before running the code. You might run it to check your answer.

**class** MyClass (object):

**def** method1(self, param\_list):

self.local\_list = []

**for** element **in** param\_list:

**if** element > 10:

self.local\_list.append(element)

**def** method2(self):

self.sum\_int = 0

**for** element **in** self.local\_list:

self.sum\_int += element

**return** self.sum\_int

inst1 = MyClass()

inst2 = MyClass()

inst1.method1([1,2,3])

print(inst1.local\_list) *# Line 1*

inst1.method1([10,11,12])

print(inst1.local\_list) *# Line 2*

print(inst1.method2()) *# Line 3*

*#inst2.method2() # Line 4*

**(a)** What output is produced by # Line 1 of the above program?

**(b)** What output is produced by # Line 2 of the above program?

**(c)** What output is produced by # Line 3 of the above program?

**(d)** # Line 4 is commented out. What result would occur if Line 4 were executed by the program. Why?

**2**. Try to answer this question before running the code. You might run it to check your answer.

**class** NewClass(object):

**def** \_\_init\_\_(self, param\_int=1):

self.the\_int = param\_int

**if** param\_int % 2 == 0:

self.parity = **'even'**

**else**:

self.parity = **'odd'**

**def** process(self, instance):

sum\_int = self.the\_int + instance.the\_int

**if** sum\_int < 0:

**return 'negative'**

**elif** sum\_int % 2 == 0:

**return 'even'**

**else**:

**return 'odd'**

**def** \_\_str\_\_(self):

**return 'Value {} is {}'**.format(self.the\_int, self.parity)

inst1 = NewClass(4)

inst2 = NewClass(-5)

inst3 = NewClass()

print(inst1) *# Line 1*

print(inst1.parity) *# Line 2*

print(inst1.process(inst2)) *# Line 3*

print(inst3.process(inst1)) *# Line 4*

**(a)** What output is produced by Line 1 of the example program?

**(b)** What output is produced by Line 2 of the example program?

**(c)** What output is produced by Line 3 of the example program?

**(d)** What output is produced by Line 4 of the example program?

**3.** Sample class describing a Person

|  |
| --- |
| **class** Person():  **def** \_\_init\_\_(self, fname, sname, address):  self.f\_name = fname  self.s\_name = sname  self.address = address  **def** change\_address(self, new\_address):  self.address = new\_address  **def** \_\_str\_\_(self):  **return** self.f\_name + **" "**+ self.s\_name + **" lives at "** + self.address    *# main*  p1 = Person(**"John"**, **"Smith"**, **"1 Pinebrook street"**)  print(p1.f\_name)  print(p1.s\_name)  print(p1.address)    p1.change\_address(**"5 Cottage Avenue"**)  print(p1) |

**(a):** Design a class to represent a **rectangle**. Some methods examples can be the rectangle area and rectangle perimeter.

**(b):** Design a class to represent a **bank account.** Some information you might want in a bank account are the IBAN, account number, available funds, a list with the last 5 transactions. You might also add methods to withdraw and deposit money.